

# ¡Bienvenidos a la presentación de Java!

Java es un lenguaje de programación popular y versátil utilizado en aplicaciones web, móviles, de escritorio y para el Internet de las cosas. Acompáñenme en este viaje para aprender más sobre Java.



# EVOLUCION DE LENGUAJES DE PROGRAMACION

B

C

POO



- ❖ Es una simplificación del lenguaje BCPL (Lenguaje de Programación Básico Combinado) capaz de almacenarse en la memoria de las computadoras de la época.
- ❖ Implementado por primera vez en 1969.
- ❖ Tenía un solo tipo de dato(palabra).
- ❖ Operaciones: suma, resta, multiplicación y división. Referencia a punteros.
- ❖ Se debía adaptar a cada arquitectura de hardware
- ❖ En 1973 se implementó el paquete de Entrada / Salida portable, que se convertiría en standard I/O (o stdio) en C.
- ❖ Se utilizo hasta los 90 en algunas maquinas Honeywell y en algunos sistemas embebidos.

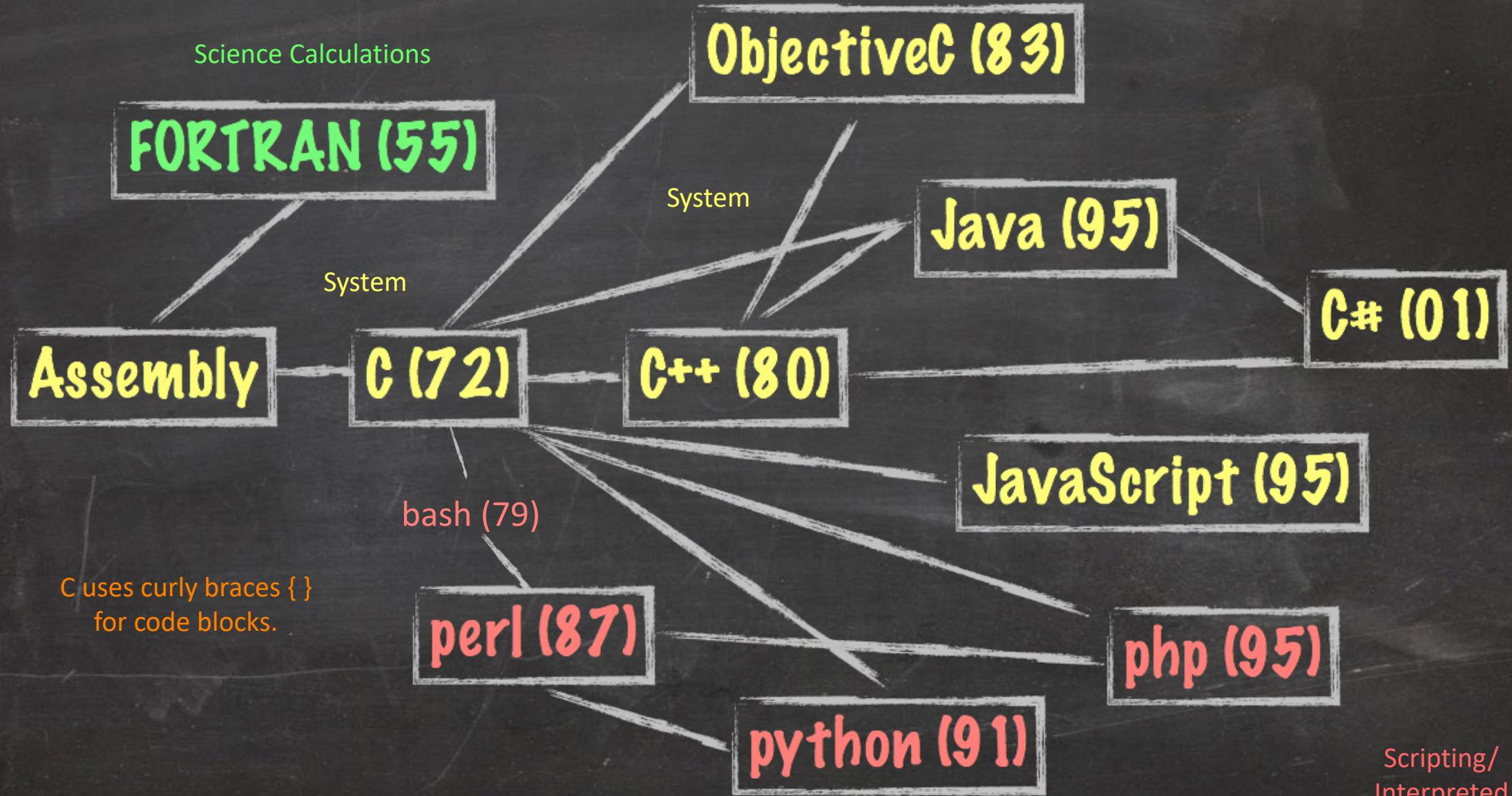


- ❖ Desarrollado entre 1969 y 1972
  - ❖ Tipos de datos primitivos int, char, float (basado en byte)
  - ❖ Facilidad de implementación y actualización de Hardware mediante core o kernell
  - ❖ Se usa en la actualidad para SO y sistemas embebidos en general
  - ❖ Se desarrolló junto con Unix
  - ❖ C++ desarrollado como forma más potente y flexible (incluye clases)

# POO

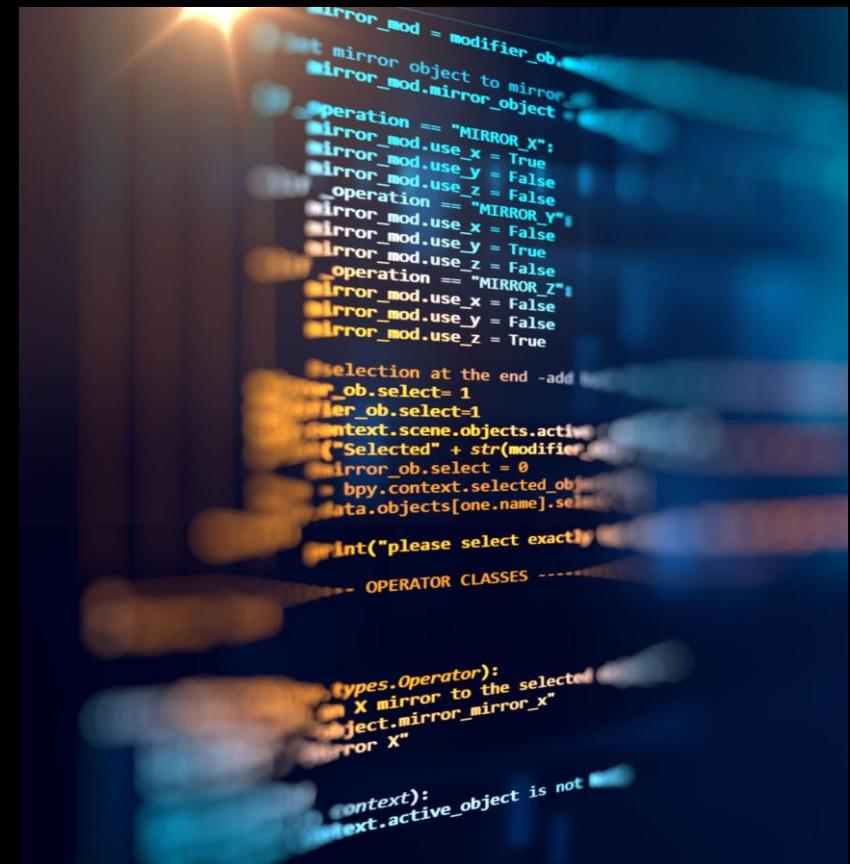
- ❖ Todo es un objeto
- ❖ Tipos primitivos (wrappers) y tipos objeto
- ❖ Adaptado a interface grafica de usuario
- ❖ Originado en la década del 60, tomó relevancia en los '90
- ❖ Basado en C
- ❖ Pilares
  - ❖ Abstracción
  - ❖ Encapsulamiento
  - ❖ Herencia
  - ❖ Polimorfismo

Science Calculations



# Java

- ❖ Creada por Sun Microsystems en 1995 y adquirida por Oracle en 2010
- ❖ Es un lenguaje compilado e interpretado a través de una máquina virtual
- ❖ Gratuito para personas y desarrolladores
- ❖ Lenguaje maduro con una gran comunidad activa
- ❖ Objetivos del lenguaje
  - ❖ Orientado a objetos
  - ❖ WORA ( Write Once Read Everywhere)
  - ❖ Robusto
  - ❖ Mejorar la experiencia de uso
  - ❖ Seguridad



# Componentes de la máquina virtual

## JRE

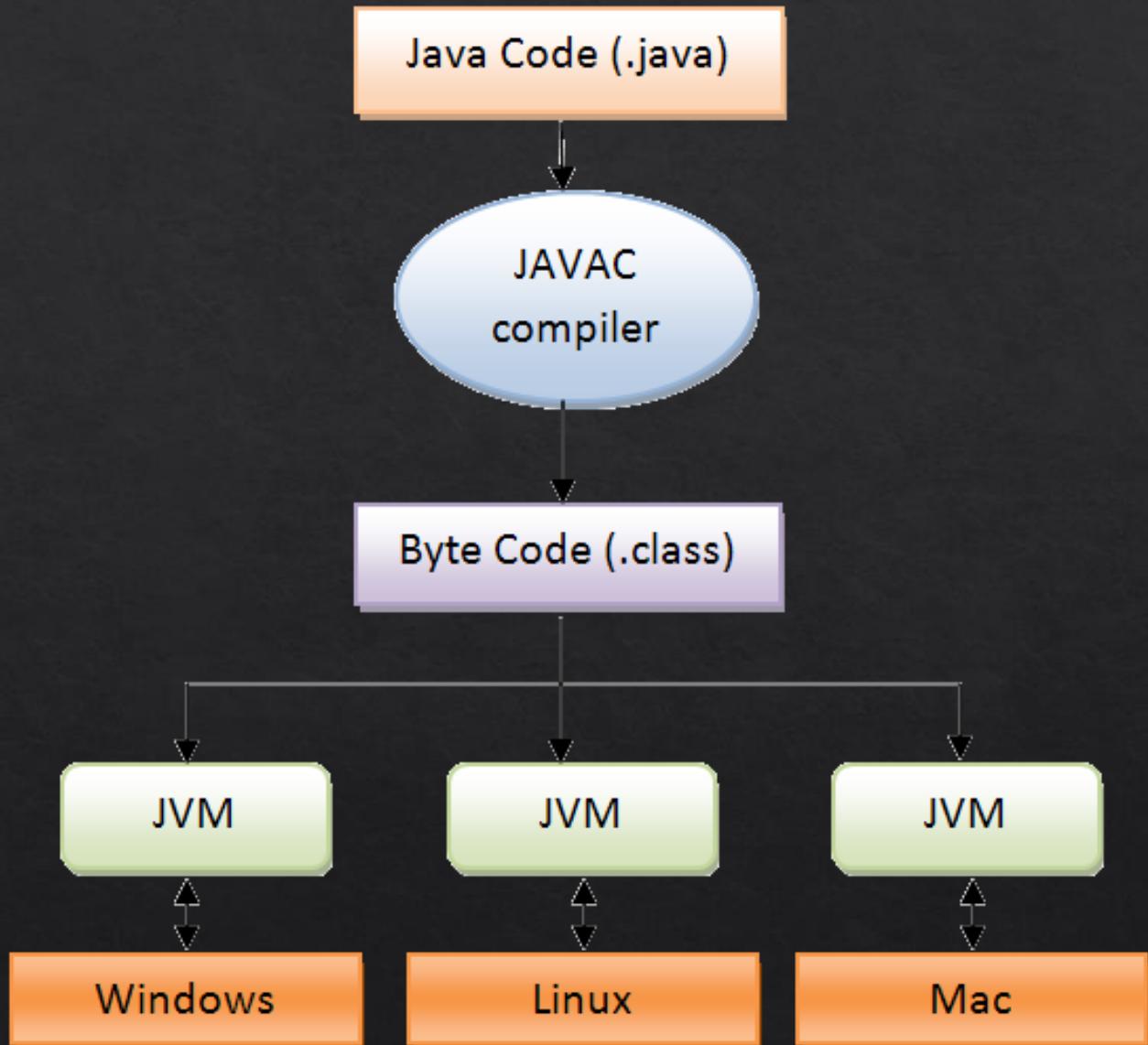
- ❖ Java Runtime Environment
- ❖ Incluye los componentes necesarios para correr aplicaciones en el SO
- ❖ Actualmente en versión 8

## JDK

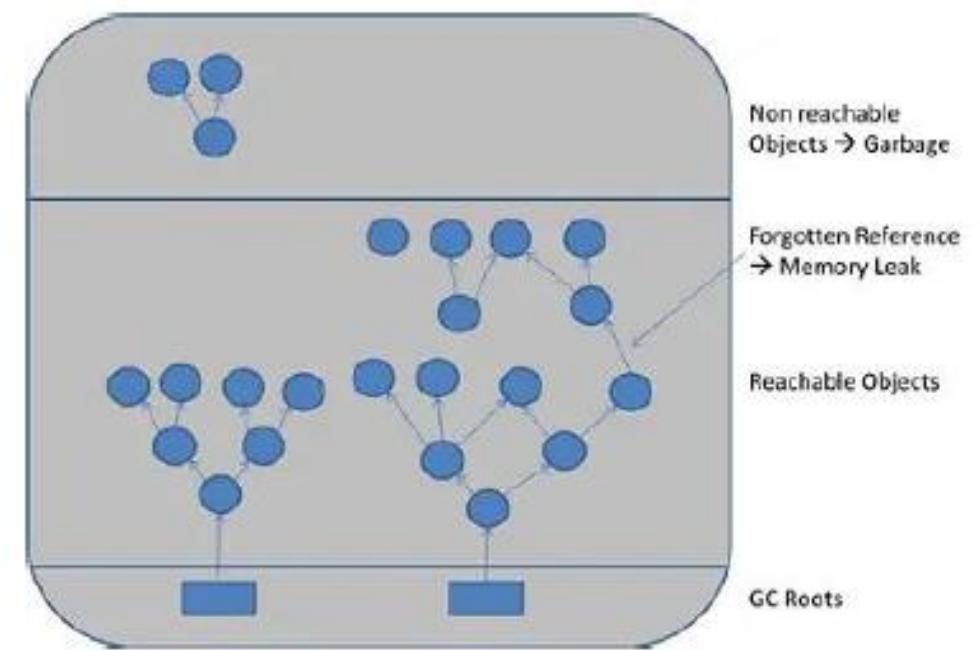
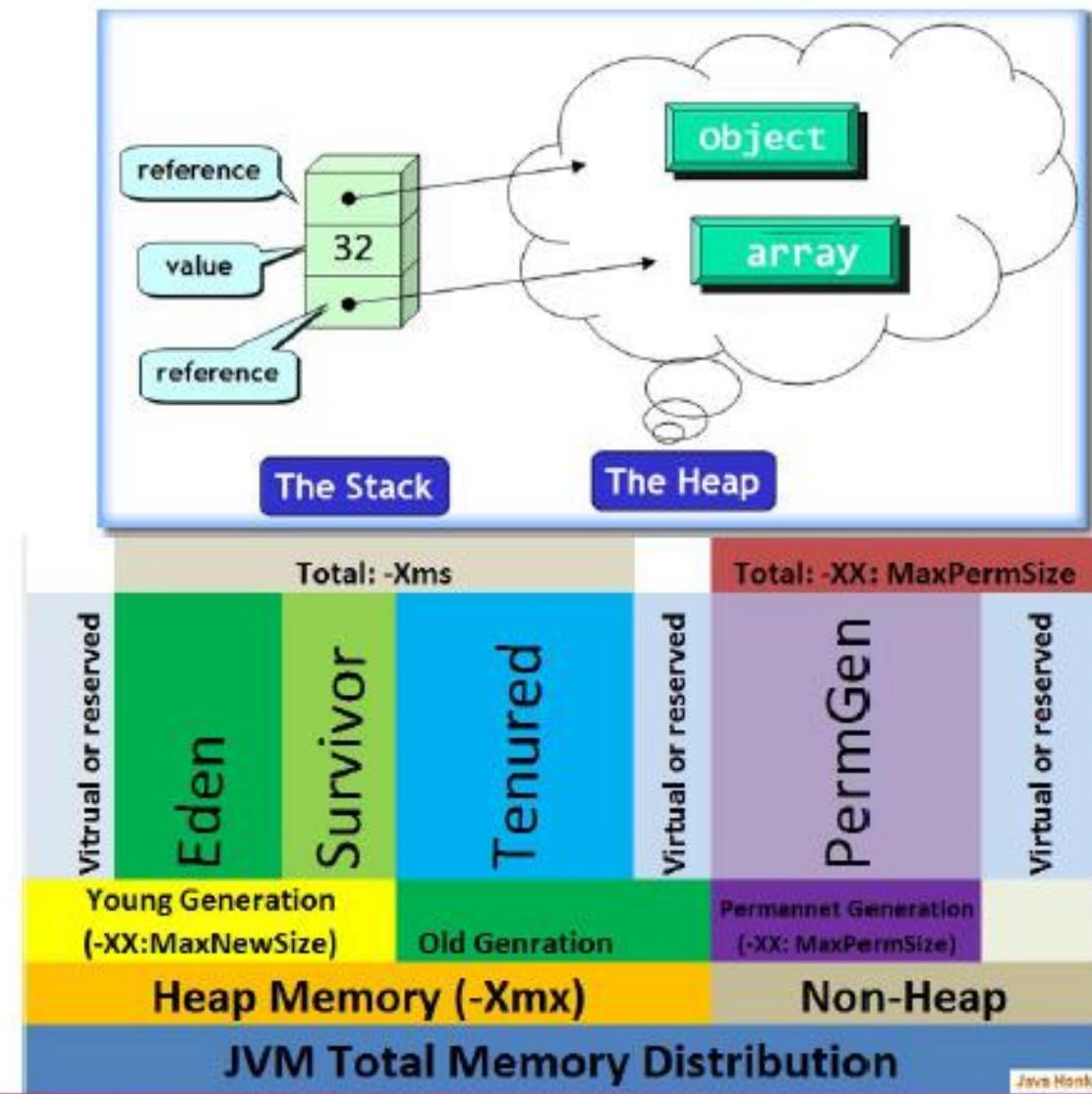
- ❖ Java Development Kit
- ❖ Incluye el JRE
- ❖ Incluye todo el entorno para desarrollar y ejecutar las aplicaciones
- ❖ Actualmente en versión 20

# JVM

- Se genera el código en .java
- Se compila en un .class
- Se ejecuta en .bin mediante JVM



# Java - Memoria y Garbage Collector



# TIPOS DE DATOS

## Primitivos

- ◊ int
- ◊ long
- ◊ float
- ◊ double
- ◊ char
- ◊ boolean
- ◊ byte
- ◊ short

## Objetos

- ◊ Tipos de la Biblioteca Estándar
  - ◊ String
  - ◊ Scanner
  - ◊ System
- ◊ Arrays
- ◊ Wrappers
  - ◊ Integer
  - ◊ Long
  - ◊ Boolean
  - ◊ Character
  - ◊ Double

# VARIABLES

## Primitivas

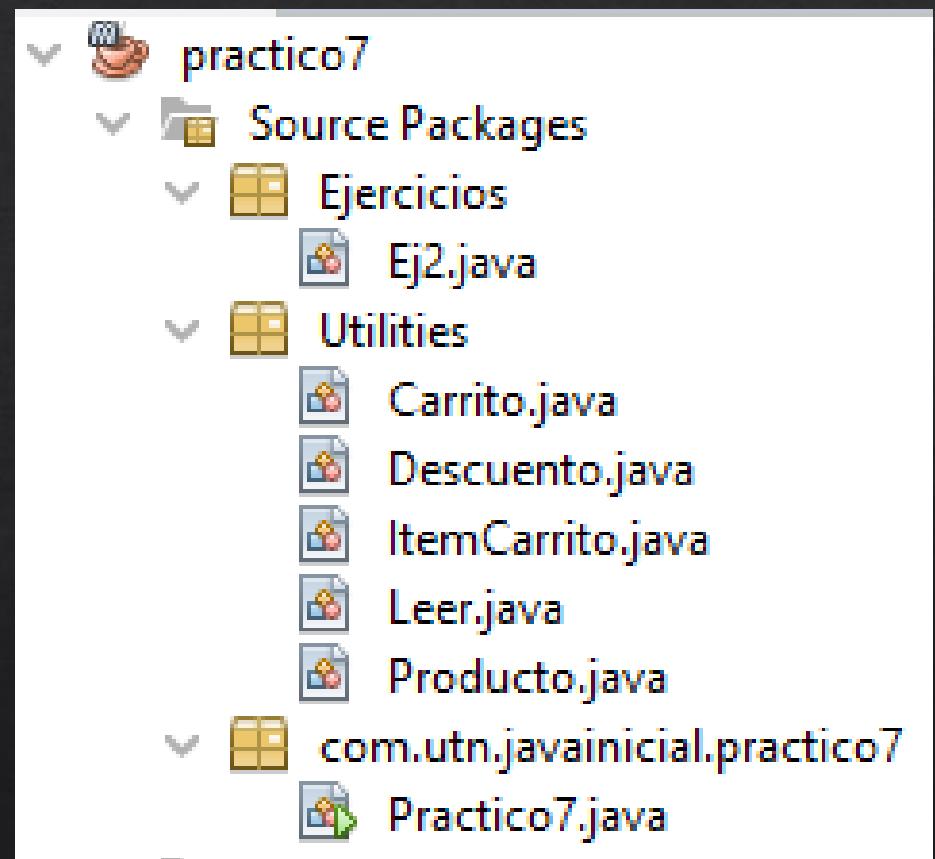
- ❖ Se guarda el valor en la variable
- ❖ Cuando se copia, se copia el valor en una nueva variable

## Objetos

- ❖ El nombre de la variable refiere a la dirección de memoria del objeto
- ❖ Cuando se copia, se copia la dirección de memoria del objeto (pointer), por lo que si se modifica la copia se modifica el original

# Estructura de datos

- ❖ Las clases se guardan en paquetes
- ❖ Los paquetes sirven para organizar clases por tipos
- ❖ Los paquetes se guardan en carpeta Source
- ❖ Puede haber otras carpetas (Test, Target)



# Composición de Clase

- ❖ Indica a que paquete pertenece
- ❖ Define atributos (encapsulamiento)
- ❖ Define constructor
- ❖ Define métodos (polimorfismo)
- ❖ Sobrescribe métodos heredados (herencia)

```
5 package Utilities;
6
7 + /**...4 lines */
8 public class ItemProducto {
9
10     private int codigo;
11     private float precioUnitario;
12     private String descripcion;
13
14     public ItemProducto(int codigo, float precioUnitario, String descripcion) {
15         this.codigo = codigo;
16         this.precioUnitario = precioUnitario;
17         this.descripcion = descripcion;
18     }
19
20     public int getCodigo() {
21         return codigo;
22     }
23     public float getPrecioUnitario() {
24         return precioUnitario;
25     }
26     public String getDescripcion() {
27         return descripcion;
28     }
29
30     @Override
31     public String toString() {
32         return "ItemProducto{" + "codigo=" + codigo + ", precioUnitario=" +
33                precioUnitario + ", descripcion=" + descripcion + '}';
34     }
35 }
```

# HERENCIA

- ❖ CLASE PADRE

- ❖ Un objeto solo puede tener un padre (extends)
- ❖ Se puede remontar el árbol de herencia hasta la clase Object
- ❖ Las clases hijas heredan los atributos y métodos de la clase padre
- ❖ Las clases hijas pueden agregar sus propios atributos y clases
- ❖ Las clases hijas pueden sobreescribir (@Override) los métodos de la clase padre (polimorfismo)

- ❖ INTERFACE

- ❖ Es una clase Abstracta (no se puede instanciar)
- ❖ Un objeto puede implementar múltiples interfaces (implements)
- ❖ Un objeto debe tener definido todos los métodos que tenga la interface implementada

# Flujo de control en Java

## 1 Condicionales

Las estructuras if y switch se utilizan para ejecutar diferentes bloques de código según una condición.

## 2 Bucles

Los bucles for, while y do-while se utilizan para ejecutar el mismo bloque de código varias veces.

## 3 Control de excepciones

El manejo de excepciones permite que las aplicaciones respondan y se recuperen de eventos inesperados.



# Errores / Excepciones

## Errores

- ❖ Son detectados por el compilador
- ❖ Normalmente evitan el funcionamiento de una aplicación desde el inicio

## Excepciones

- ❖ Puede no ser detectado por el compilador
- ❖ Se generan cuando el programa tiene que procesar un dato inesperado
- ❖ Puede ser atrapado y manejado con un bloque try/catch (RuntimeException)
- ❖ Se pueden generar excepciones personalizadas (Throwable)

# Excepciones Comunes

Java y algunas librerías base, ya establecieron excepciones comunes y algunas de ellas deberían ser utilizadas o heredadas antes de crear otras.

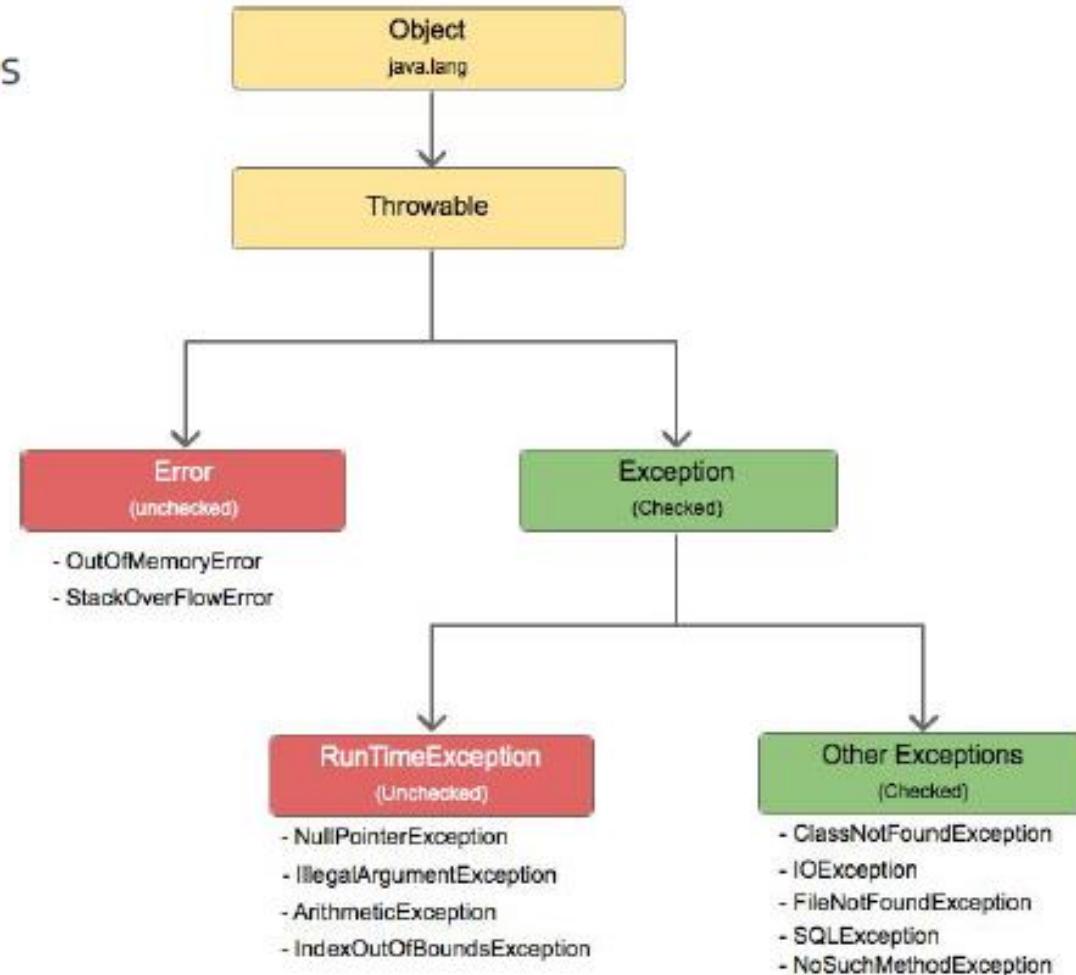
No chequeadas

- NullPointerException
- DivisionByZeroException
- IllegalStateException

Chequeadas

- FileNotFoundException
- IOException

....



# Manejo de Excepciones

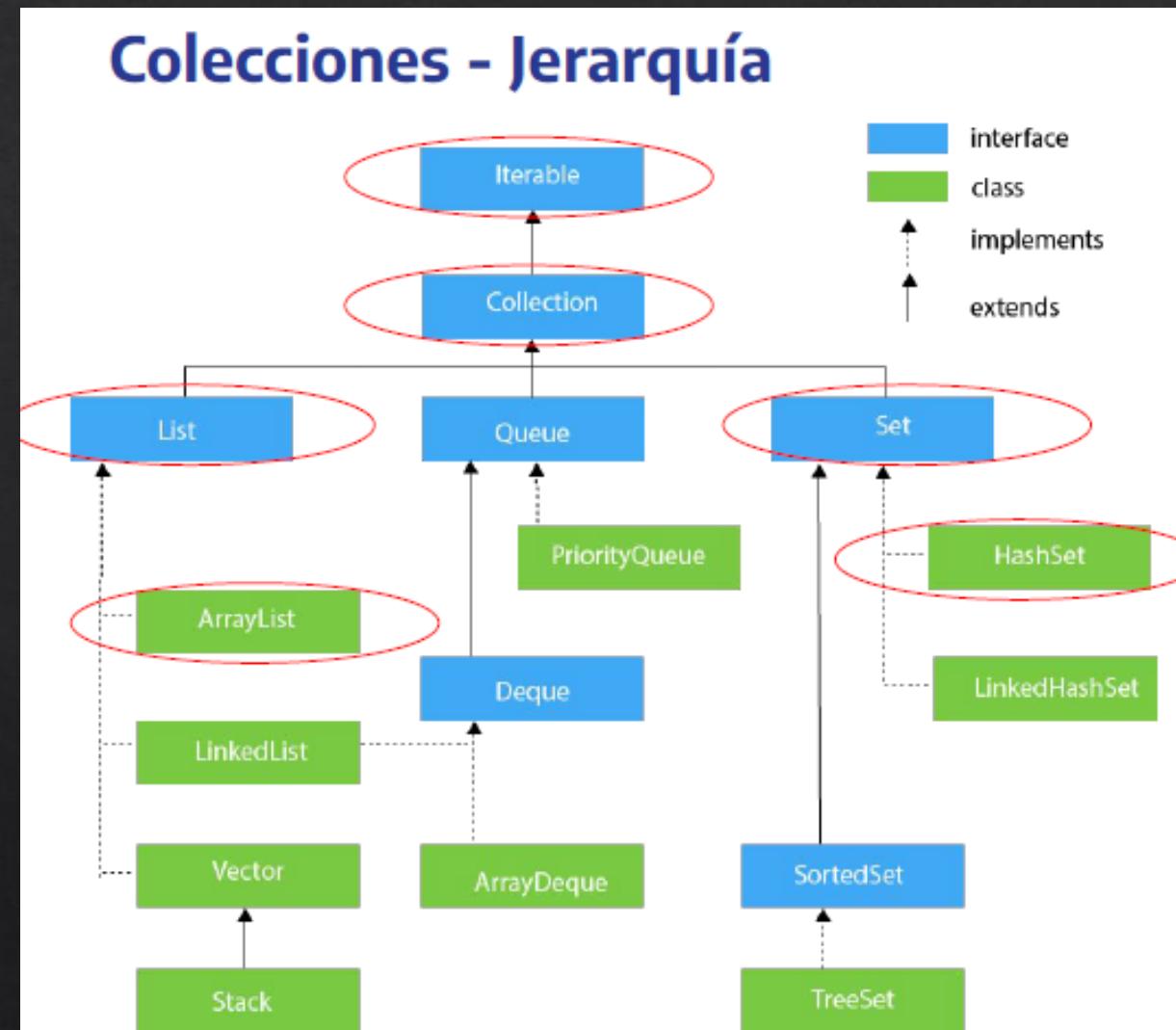
- ❖ Evita que el programa se rompa por un dato incorrecto o inexistente
- ❖ Normalmente se usa para leer datos o conexiones

```
Path path = Paths.get(first:ruta);
List<String> fh = new ArrayList();
try {
    fh = Files.readAllLines(path);
} catch (Exception e) {
    System.out.println("No se pudo abrir archivo!\n/* + e*/");
}

if (!fh.isEmpty()) {
    return fh.toString();
} else return null;
```

# Colecciones

- ❖ Similares a los arrays[]
- ❖ Tamaño dinámico
- ❖ Almacenan diferentes tipos de datos
- ❖ Tienen métodos heredados
- ❖ Mas usados:
  - ❖ ArrayList
  - ❖ LinkedList
  - ❖ HashMap



# PERSISTENCIA DE LA INFORMACION

- ❖ Bases de datos
  - ❖ Bases de datos relacionales (SQL)
    - ❖ MySQL
    - ❖ PostgreSQL
    - ❖ SQLite
    - ❖ MariaDB
  - ❖ Bases no Relacionales
    - ❖ MongoDB
  - ❖ XML o JSON
- ❖ Archivos
  - ❖ Texto
  - ❖ CSV
  - ❖ Imágenes / Multimedia
  - ❖ Binario

# GESTION DE DEPENDENCIAS

- ❖ Optimizan el mantenimiento y lectura de una Aplicación
- ❖ Facilidad de gestión de librerías (jar)
- ❖ Facilita el control de versiones
- ❖ Automatizacion de tareas de compilación y ejecución



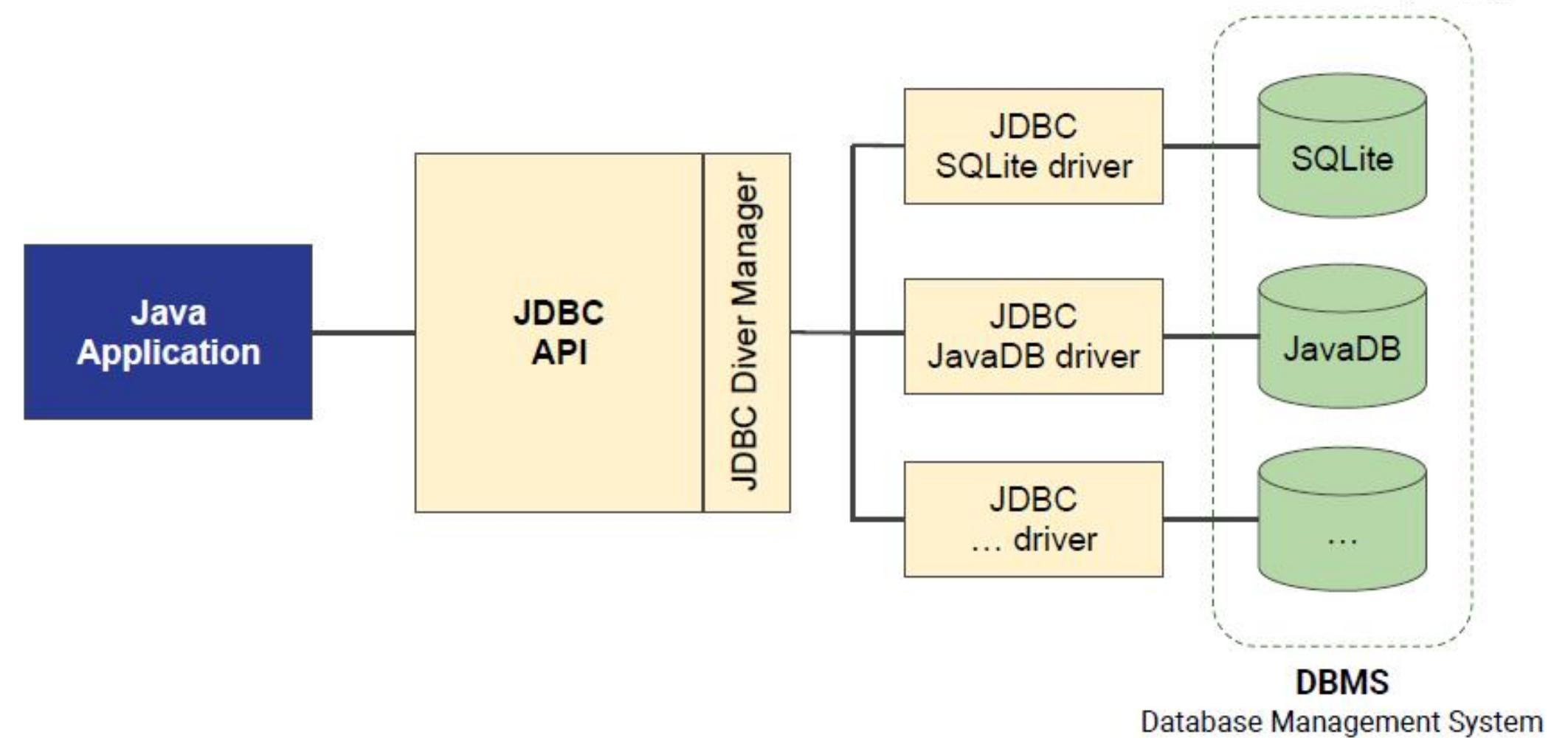
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.utn.javaInicial</groupId>
    <artifactId>practico8</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <exec.mainClass>com.utn.javainicial.practico8.Practico8</exec.mainClass>
    </properties>
    <dependencies>
        <!-- https://mvnrepository.com/artifact/junit/junit -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```

# JDBC

- ❖ Java DataBase Conectivity es una API (Application Programming Interface)
- ❖ Conecta la aplicación con una base de datos
- ❖ Utiliza un Driver para conectar a la base
- ❖ Permite CRUD mediante statements
- ❖ Implementación:
  - ❖ Registrar Driver - Crear Conexión - Crear Sentencia - Ejecutar Query - Cerrar Conexion

```
try {  
    String jsonText = om.writeValueAsString( value:alumno.getListaMaterias());  
    String sql = "INSERT INTO alumnos(legajo, nombre, materias_aprobadas) VALUES (?, ?, '"  
    + jsonText + "')";  
    PreparedStatement st =con.prepareStatement(sql);  
    st.setInt( parameterIndex: 1,  #:legajo);  
    st.setString( parameterIndex: 2,  #:nombre);  
    st.execute();  
    st.close();  
    con.close();  
} catch (Exception e) {  
    System.out.println("Error! " + e);  
}
```

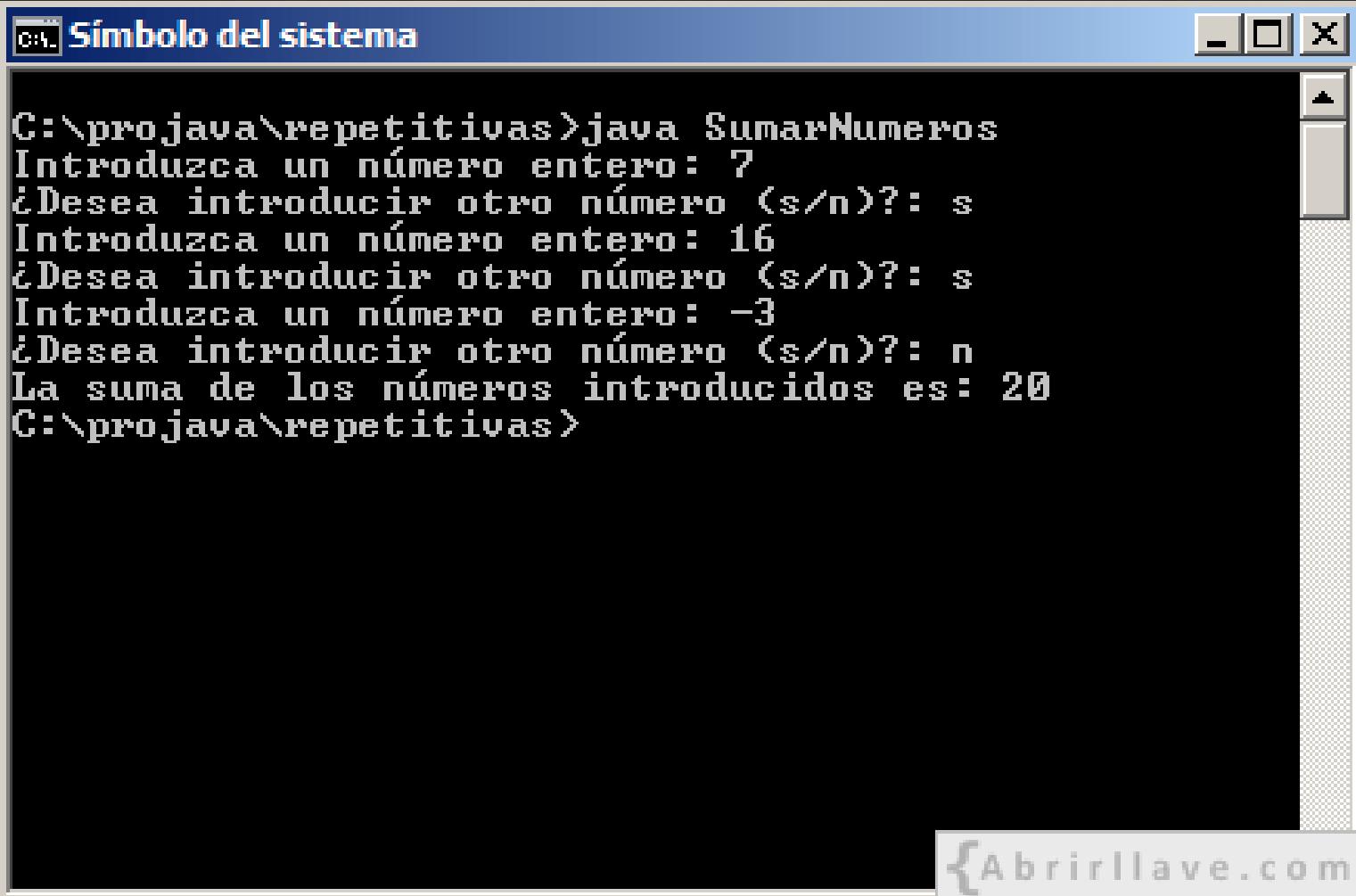
# JDBC - API



# APLICACIONES

- ❖ Consola
  - ❖ Aplicaciones de solo texto (sin entorno gráfico)
- ❖ Desktop Apps
  - ❖ Winforms
  - ❖ Aplicaciones Instalables
- ❖ Web Apps (web2)
  - ❖ APIs
  - ❖ Aplicaciones ejecutables desde el navegador

# CONSOLA



C:\projava\repetitivas>java SumarNumeros  
Introduzca un número entero: 7  
¿Desea introducir otro número (s/n)?: s  
Introduzca un número entero: 16  
¿Desea introducir otro número (s/n)?: s  
Introduzca un número entero: -3  
¿Desea introducir otro número (s/n)?: n  
La suma de los números introducidos es: 20  
C:\projava\repetitivas>

# DESKTOP

Sistema 1.0

Archivo Ayuda

Formulario

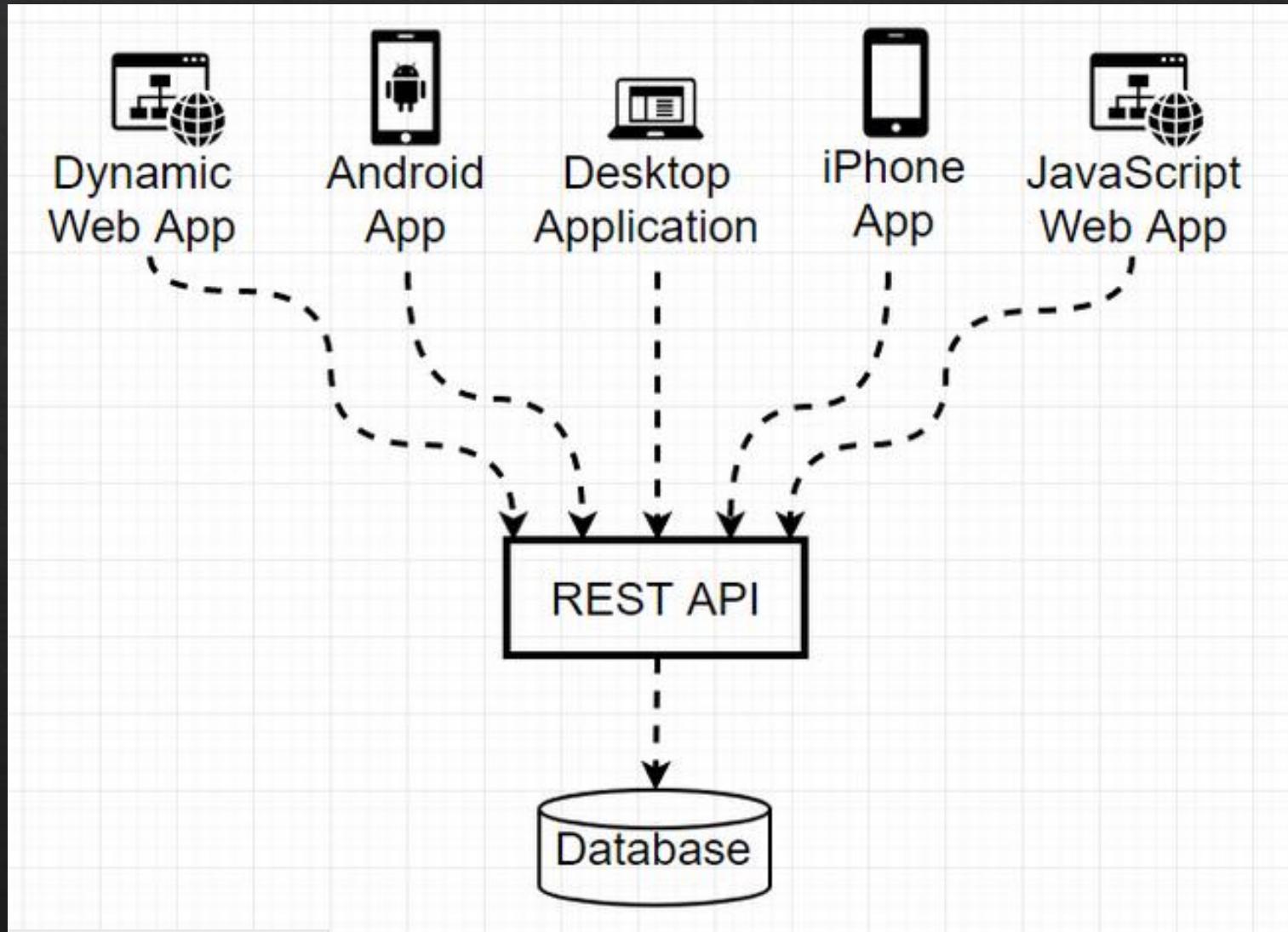
Nombre	Pedro
Apellido	Martinez
Pais Natal	Chile ▾

Guardar Cancelar Limpiar Datos

Escuela de Informática 2013 - UNPA UACO

The screenshot shows a desktop application window titled "Formulario". The window has a title bar with a minimize button, maximize button, and close button. Below the title bar is a menu bar with "Archivo" and "Ayuda" options. The main area of the window contains three input fields: "Nombre" with the value "Pedro", "Apellido" with the value "Martinez", and "Pais Natal" with the value "Chile" and a dropdown arrow. At the bottom of the window are three buttons: "Guardar", "Cancelar", and "Limpiar Datos". The overall interface is simple and functional.

# API



# WEB APP

Image Upload Analysis

localhost:8080

## Image Upload Analysis

Upload an image:

stanley.jpg

<https://happycoding.io/>

localhost:8080/image-analysis

localhost:8080/image-analysis

Here's the image you uploaded:



Here are the labels we extracted:

- Cat 0.9858267
- Felidae 0.9510577
- Small to medium-sized cats 0.9379458
- Whiskers 0.91683894
- Leaf 0.8793513
- Carnivore 0.75382805
- Snout 0.7342774
- Kitten 0.70020294
- Fawn 0.646203
- Plant 0.58547497

# RECURSOS

- ❖ Argentina Programa
- ❖ Codo a Codo
- ❖ FreeCodeCamp
- ❖ Tutoriales Web
- ❖ Alura ONE
- ❖ Tecnicaturas
  - ❖ UTN
  - ❖ ISPC
- ❖ ¡Mucha Practica!



# ¡¡¡GRACIAS!!!

<https://github.com/zalotores/SeminarioJava.git>

zalotores@gmail.com

